Express Mail No. EL610089397US

IBM DOCKET: ROC9-2000-0142
WHE DOCKET: IBM-158

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: **TIME-MULTIPLEXING DATA BETWEEN ASYNCHRONOUS CLOCK DOMAINS WITHIN CYCLE SIMULATION AND EMULATION ENVIRONMENTS**

APPLICANTS: **Thomas Michael Gooding, Roy Glenn Musselman, Robert N Newshutz and Jeffrey Joseph Ruedinger**

ASSIGNEE: **International Business Machines Corporation**

Wood, Herron & Evans, L.L.P.
2700 Carew Tower
Cincinnati, Ohio 45202
513-241-2324

SPECIFICATION

TIME-MULTIPLEXING DATA BETWEEN ASYNCHRONOUS CLOCK DOMAINS WITHIN CYCLE SIMULATION AND EMULATION ENVIRONMENTS

Field of the Invention

The invention is generally related to simulation and emulation of integrated and other electronic circuit designs. More specifically, the invention is generally related to the communication of data within a hardware-based cycle simulation or emulation environment.

Background of the Invention

As semiconductor fabrication technology advances, designers of integrated circuits and electronic circuits incorporating the same are able to integrate more and more functions into individual integrated circuit devices, or chips. As such, electronic designs that once required several integrated circuits electrically coupled to one another on a circuit board or module may now be integrated into fewer integrated circuits, thereby increasing performance and reducing cost.

With increases in circuit complexity, however, the processes of designing and testing circuit designs become increasingly complex and time consuming. As a result, computers have become increasingly important in automating the design and testing of circuit designs.

An important step in the development of a complex electronic system is that of verification, which is used to verify the functional operation of a circuit design. Traditionally, hardware circuit designs have been designed on a computer at a relatively high level of abstraction, typically in a hardware definition language such as VHDL or Verilog. Software tools, known as compilers, are then used to generate

simulation models for the designs that can be executed on a logic simulator computer program to simulate the reactions of such circuit designs to various input conditions. By simulating the functional operation of a circuit design, potential errors or faulty logic can be identified and corrected in the high level design. Simulation is then rerun until the circuit design functions as desired.

However, with the increasingly complex nature of many circuit designs, software-based simulation is often too time consuming and inefficient. As a result, a significant amount of development effort has been directed toward hardware-based verification environments such as cycle simulators and logic emulators (hereinafter jointly referred to as "hardware-based emulators"). Both cycle simulation and logic emulation of a circuit design are often performed using a massively parallel hardware-based emulator incorporating hundreds or thousands of "emulation processors" that are used to emulate, in hardware, the various functional components of a circuit design. The emulation processors can be specifically designed to efficiently emulate various functional components, and thus permit the emulation of potentially millions of logic gates in substantially less time than would be required for software-based simulation.

Cycle simulation, for example, is used to verify the functionality of a circuit design by calculating the outputs of circuit components at clock edges, with typically only two logic states (binary 1 and 0) computed for each component output.

Related to cycle simulation is in-circuit logic emulation, which verifies the operation of a circuit design within an overall electronic system. For example, in-circuit logic emulation may be used to verify the proper interaction between a circuit design and a real-world target hardware system with which the eventually-manufactured product will operate. Moreover, in-circuit logic emulation may be used to verify a hardware circuit design running custom software, a process known as "coverification". Thus, with in-circuit logic emulation, system-wide integration, testing and verification can be substantially simplified.

One exemplary type of hardware-based emulation environment is the ET3.5 emulation system from International Business Machines Corporation. The ET3.5 system includes potentially thousands of emulation processors distributed among

multiple interconnected logic boards, with each logic board including sixty-five processor chips, and with each processor chip including sixty-four emulation processors. Emulation processors can individually emulate hundreds of logic gates, thereby permitting millions of logic gates to be emulated at any given time.

With a hardware-based emulator, a logic design to be emulated is typically in the form of a gate-level model that has been compiled from a high-level language. The compiled model breaks up each clock cycle (also referred to as an evaluation cycle) into a series of evaluation "steps". The evaluation steps are typically executed in-order on each emulation processor, and are repeated during each evaluation cycle.

Given the highly parallel nature of most circuit designs, the ability to communicate information between emulation processors during logic emulation is often extremely important for accurately and efficiently verifying a logic design. To support such inter-processor communication in the ET3.5 system, for example, each processor chip is connected to a pair of connectors capable of interfacing the chip with other chips disposed on the same or different logic boards within the system. Each processor on the chip has a dedicated input pin (XBI) and output pin (XBO), and thus thirty-two processors use each connector. Processors are capable of reading and writing to their respective XBI/XBO pins during any evaluation step, and as a result, it is possible to time-multiplex multiple data signals between two chips using the XBI/XBO pins.

While the use of the aforementioned connectors between processor chips often provides adequate connectivity between processors, the interconnection mechanism is typically not capable of supporting asynchronous communications between processors with the same degree of connection density.

Asynchronous communications, for example, are required in a logic system whenever two electronic components are operating on different system clocks. All of the logic in a circuit design that operates under the same clock is typically referred to as a "clock domain." Particularly in circuit designs incorporating multiple logic boards, as well as in designs that interface with target hardware systems, it is common for multiple clock domains to be utilized. While many logic emulators support the ability to emulate multiple clock domains, communicating signals between clock

domains is problematic, since the receiver of an asynchronous signal has no clock relationship to the source, and thus cannot decode time-multiplexed data over a given signal path. As a result, conventional logic emulators have required that no more than one signal be communicated over a particular signal path between asynchronous clock domains. Otherwise, time-multiplexed signals could be misinterpreted and cause the generation of incorrect emulation results.

When only a few sets of signals are communicated between asynchronous clock domains in a circuit design, the limitation of one signal per signal path is not particularly burdensome. However, in many complex systems, thousands of signals may be required to be passed between asynchronous clock domains. With conventional logic emulators such as the aforementioned ET3.5 system, it has been found that tens or hundreds of cables may be required to communicate all necessary data between asynchronous clock domains. In many instances, these cables are very expensive, thus adding significant cost to a hardware-based emulation system. In addition, the more cables in a system, the more difficult the system is to administer, and the greater likelihood that a cable may fail or not be connected properly. Furthermore, connectivity limitations may decrease the number of logic gates that may be emulated in a given system, thus requiring a more complex, and thus more expensive, system to adequately emulate a particular design.

Therefore, a significant need continues to exist in the art for a manner of facilitating the communication of data signals between asynchronous clock domains in a hardware-based emulation environment. In particular, a need has existed for overcoming the limitation against time-multiplexing data across signal paths between asynchronous clock domains during emulation and/or simulation of a circuit design.

Summary of the Invention

The invention addresses these and other problems associated with the prior art by providing an apparatus and method that utilize a buffer interposed in a common signal path between asynchronous clock domains in a hardware-based emulation environment to manage the communication of time-multiplexed data signals between the asynchronous clock domains during hardware-based emulation. The buffer is effectively used to latch each data signal communicated across the common signal path so that the clock domain that receives the signals can retrieve each such signal at appropriate points in the receiver clock domain's evaluation cycle.

In the illustrated embodiments, independently-controlled write/read pointers are typically maintained in a buffer control circuit to independently address the buffer for the transmitter and receiver sides of an asynchronous communication path. Locations in the buffer are associated with specific steps in the evaluation cycles of each of the transmitter and receiver clock domains, and the write/read pointers are managed to respectively write and read data to and from the locations in the buffer based upon the current evaluation steps being performed within the respective evaluation cycles of the transmitter and receiver clock domains.

As such, time-multiplexed data that is transmitted by a transmitter clock domain can be effectively routed and decoded by a receiver clock domain over the course of the receiver clock domain's evaluation cycle, thereby substantially increasing the signal-carrying capacity of the signal path. As a result, the cabling requirements between asynchronous clock domains may be substantially reduced, thus increasing system reliability and simplicity, and decreasing costs.

These and other advantages and features, which characterize the invention, are set forth in the claims annexed hereto and forming a further part hereof. However, for a better understanding of the invention, and of the advantages and objectives attained through its use, reference should be made to the Drawings, and to the accompanying descriptive matter, in which there is described exemplary embodiments of the invention.

Brief Description of the Drawings

FIGURE 1 is a block diagram of a hardware-based emulation system incorporating an asynchronous buffer card consistent with the invention.

FIGURE 2 is a block diagram of the asynchronous buffer card of Fig. 1, shown interfaced with emulation processor chips from a pair of asynchronous clock domains.

FIGURE 3 is a table illustrating an exemplary communication of time-multiplexed data between asynchronous clock domains using the hardware-based emulation system of Fig. 1.

Detailed Description

Turning to the Drawings, wherein like numbers denote like parts throughout the several views, Fig. 1 illustrates a hardware-based logic emulation system 10 consistent with the invention. System 10 includes a hardware-based logic emulator 12, which is used to emulate a circuit design stored in a database 14. The circuit design is typically designed and modeled in a high-level hardware definition language such as Verilog or VHDL. The logic emulator 12 operates on a gate-level compiled model of the circuit design, which may be generated, for example, on a host workstation 16 that executes a compiler 18. The host workstation may also be used to design or perform other Electronic Design Automation (EDA) steps during the development, testing, verification and/or validation of a circuit design.

Logic emulator 12 may be based upon any number of suitable hardware-based logic emulation or simulation environments, e.g., the ET3.5 emulation environment from International Business Machines Corporation. Typically, logic emulator 12 includes a plurality of logic boards 20, each of which including a plurality of emulation processor chips 22. Moreover, each chip 22 typically includes a plurality of emulation processors for emulating a compiled circuit design model. In the ET3.5 environment, for example, each emulation processor chip includes sixty-four emulation processors, with each logic board including 65 emulation processor chips disposed thereon. It will be appreciated, however, that other hardware-based logic emulator architectures may be used in the alternative.

Inter-processor communication in logic emulator 12 is typically handled by a plurality of connectors 24 interfaced with chips 22. In the illustrated embodiment, for example, Micropax connectors from Berg Electronics may be utilized to interface emulation processor chips with one another. Each chip 22 includes a pair of Micropax connectors, each connector including sixty-four signal pins, with thirty-two designated as input pins and thirty-two designated as output pins. Thus, with the pair of connectors allocated to each chip, each emulation processor on a chip is allocated one input pin (designated XBI) and one output pin (designated XBO) between the pair of connectors for the chip. For synchronous communications, Micropax cables are typically routed between connectors associated with different chips. Typically, the

connectors are routed between different logic boards, although chips located on the same logic board may also be interfaced in the same manner. Moreover, the connectors may be interconnected with other types of electronic components within logic emulator 12, e.g., memory boards, input/output boards, data capture boards, etc.

As discussed above, for synchronous communications, the use of direct cables between chips may permit multiple data signals to be transmitted over common signal paths in a Micropax cable using time-multiplexing, given that both the receiver and transmitter chip are running in the same clock domain. Circuit designs that execute within the same clock domain are typically compiled at the same time, and include the same number of emulation evaluation steps per evaluation cycle. Thus, so long as the number of signals that need to be transmitted between two chips within a given evaluation cycle is less than the total number of evaluation steps, one signal path may be used to transmit all of the signals each evaluation cycle. In certain instances, more data than is allowed by the number of evaluation steps may need to be transmitted, whereby it may be desirable to allocate more than one pin to a particular processor on a given chip.

However, for asynchronous communications, where processors on different chips are emulated in different clock domains, time-multiplexed communications over a single signal path using a direct cable is often not possible, particularly if the respective clock domains utilize different numbers of evaluation steps per evaluation cycle. This is based upon the fact that if the clock domains have different evaluation steps, it would not be possible at the receiver end to be certain which step data is being sent over a single signal path. In addition, the start of execution of evaluation steps may occur at different times if the domains are each synchronized to independently running target systems.

To address this limitation with conventional logic emulators, an asynchronous buffer card 30 is interposed between chips 22 that reside in separate clock domains (e.g., clock domains A and B illustrated in Fig. 1). The asynchronous buffer card may be coupled to each chip via Micropax compatible cables, or in the alternative, the buffer card may include a Micropax connector for a daughter board-type connection to one of the Micropax connectors associated with a chip, with a Micropax-compatible

cable utilized to interface the buffer card with another chip. The respective connections between buffer card 30 and chips 22 are illustrated at 32, and should be considered to be capable of being implemented as cables, connectors, and other forms of interconnects known in the art. Moreover, it will be appreciated that the buffer card may be utilized to interface chips on the same logic board, or on separate logic boards, consistent with the invention. Further, the buffer and buffer control circuitry may not be disposed on a card, but may be disposed in other components, and may be integrated with other components (e.g., in a target system, in a processor chip, on a logic board, etc.).

Moreover, as illustrated by target hardware block 34, it may also be desirable to utilize an asynchronous buffer card to interface a chip with a target hardware system, e.g., for in-circuit logic emulation. As a result, rather than interfacing two emulation processor chips to one another, an emulation processor chip may be interfaced with a real-world hardware system with which the emulated circuit design is being interfaced. The interconnection 36 between target hardware 34 and buffer card 30 may be via a cable or other interconnect, or in the alternative, the functionality of card 30 may be incorporated into the external hardware device, and connected to the logic emulator via suitable Micropax-compatible cabling.

It will be appreciated that other logic emulators and emulation systems may be utilized in the alternative. Moreover, the principles of the invention may apply to either or both of cycle simulation and in-circuit logic emulation, or a system may be capable of addressing both types of verification environments. For this reason, the generic terminology "emulation" is used hereinafter to refer to both cycle simulation and logic emulation.

Moreover, different logic emulator architectures may be used in the alternative, with the partitioning of emulation processors among chips and boards differing from that illustrated in Fig. 1. Furthermore, additional electronic components, e.g., I/O cards, memory, data capture cards, etc., may also be incorporated into logic emulator 12 consistent with the invention. Furthermore, other interconnects between emulation processors, as well as connectors other than

Micropax connectors, may also be used in the alternative. Thus, the invention is not limited to the particular environment discussed herein.

Asynchronous buffer card 30 is illustrated in greater detail in Fig. 2. In particular, a unidirectional communication pathway in buffer card 30 is illustrated, and is suitable for transmitting data generated by a processor 40 in clock domain A, to a processor 42 in clock domain B. It will be appreciated that corresponding circuitry for communicating from clock domain B to clock domain A may also be used in the alternative. Moreover, card 30 may include multiple connectors to interface multiple emulation processor chips to one another, including environments where more than two clock domains are used.

Typically, when a multi-clock domain hardware design is compiled, the separate clock domains are compiled into separate models. At compile time, the compiler typically determines which value will be output or input by a processor at what step in an evaluation cycle. To support time-multiplexing of signals across a given signal path between clock domains, therefore, each processor on each chip 22 typically includes a processor data stack including one or both of an output stack and an input stack (e.g., output stack 44 for processor 40 and input stack 46 for processor 42). Each slot in the associated stack 44, 46 is associated with a particular step in an evaluation cycle. For example, it may be desirable to provide up to 256 steps in an evaluation cycle, whereby each stack 44, 46 would include a maximum of 256 slots for each processor on the chip.

Fig. 2 illustrates, as an example, the interface between a pair of processors 40, 42 located in clock domains A and B, where the first processor 40 is coupled to an XBO(0) pin on connector 24 for its chip 22, with processor 42 in clock domain B coupled to the corresponding XBI(0) pin on the respective connector 24 on its chip 22. The XBO and XBI pins for the other processors on each chip 22 are designated as XBO(1-31) and XBI(1-31). These processors are not shown separately in Fig. 2.

As an example of the use of separate compiled models, Fig. 2 also illustrates that the circuit design in clock domain A is modeled using 105 evaluation steps, while the circuit design modeled in clock domain B includes 197 evaluation steps. The total

number of steps in an evaluation cycle corresponds to one real-world clock cycle for the model, and the number of steps can vary for different circuit design models.

Thus, during each cycle of a logic emulation process, processor 40 in clock domain A outputs up to 105 data signals to stack 44, while processor 42 in clock domain B reads the contents of the input stack 46 during its respective 197 evaluation steps.

If the processors 40, 42 in clock domain A and clock domain B were within the same clock domain and therefore used the same number of evaluation steps per real-world cycle, the contents of output stack 44 could be simply asserted on the XBO(0) signal path defined between the chips, with the asserted XBO(0) pin connected directly to the XBI(0) pin on the other chip, so that the value stored in any slot in output stack 44 is fed directly to the corresponding input slot in input stack 46.

For asynchronous communications, however, asynchronous buffer card 30 is interposed into the signal path between the matching XBO and XBI pins on the respective connectors 24 of the chips 22 in clock domains A and B. To provide such an interface, a dual port memory array, buffer or discrete bank of registers (for the case of multiplexing on one end and discrete signals on the other end (e.g., a target system)) 50 is interposed between the XBO pins and XBI pins, with the interconnects 32 between connectors 24 being coupled to data in (write) 50A and data out (read) 50B ports on the dual port array. Thus, it may be seen that the XBO/XBI connection represents a common signal path through which time-multiplexed signals may be communicated by a transmitter with the clock domain buffer 50 interposed in that path to latch the signals for later capture by the receiver clock domain.

In the illustrated embodiment, 32 XBO pins are implemented on a particular connector 24, as are 32 XBI pins on the other connector for a chip. Thus, to latch the 32 pins in a unidirectional communication, buffer 50 is 32 bits in width, with each of the 32 bits mapped to a particular XBO/XBI set of pins.

In the illustrated embodiment, a multi-port array such as a dual port array is utilized for buffer 50 to permit concurrent reads and writes to the buffer. Addressing of read and write operations is managed by a pair of counters 52, 54, which respectively operate as write and read control circuits, and which are respectively

driven by the associated clock domains at the output and input sides of the signal path. In the illustrated embodiment, each clock domain provides a step pulse for the respective clock domain during each step of an evaluation cycle. Moreover, since each clock domain may include a different number of evaluation steps, a step start signal is provided to each counter 52, 54 to reset the counter at the beginning of each evaluation cycle.

With this configuration, therefore, read and write pointers into the dual port array 50 are independently managed by the respective clock domains. This effectively creates a latch for each time-multiplexed signal over a particular XBO/XBI signal path, which keeps the value stable at all times so the receiver can access it whenever appropriate.

As shown in Fig. 2, multiple arrays 50 may be used in a buffer. For example, a second array may be used for the complementary transmissions from processor 42 to processor 40. Otherwise, other arrays may be used to support additional communication signal paths between the chips, or between additional chips and/or target systems. It should also be appreciated that, for a complementary array, counter 54 could be used to drive the write pointer for the complementary array, with counter 52 driving the read pointer on behalf of the complementary array.

Next, Fig. 3 illustrates an exemplary emulation operation between a pair of clock domains "A" and "B" using the asynchronous buffer card of Fig. 2. In this example, clock domain A includes six evaluation steps per evaluation cycle, while clock domain B takes four evaluation steps per real-world clock cycle. It also assumes that data is being transmitted from clock domain A to clock domain B across a one-bit wide common signal path, with a one-bit signal BUSB(9) being transmitted in step 1, and with one-bit ENABLE and FLUSH signals being transmitted during steps 2 and 3 of each evaluation cycle. Fig. 3 illustrates, for clock domain A, the time in units of evaluation steps, whether or not the step pulse signal is active, the current step in an evaluation cycle, and the value output on the XBO pin for the transmitting processor (a value of "X" is a don't-care). Likewise, for clock domain B, the status of the step pulse line, the step number, and the value read (i.e, the value asserted on the corresponding XBI pin), are shown.

For the asynchronous buffer card, the values asserted at the array input and output (i.e., the data in and data out lines to the array) are illustrated, as are the contents of six locations in the array, designated [0] - [5]. Moreover, the current value stored in the counters associated with clock domain A and clock domain B (counters 52 and 54) are also displayed, as is the indication of a counter reset in association with the values of "0" for each counter. (It is assumed for this example that any time a step start signal is asserted, the appropriate counter will go to a value of "0".)

Generation of the step start and step pulse signals may be performed in a number of manners consistent with the invention. For example, the step pulse signal may be generated from a FastCLK signal generated by the appropriate clock domain. The FastCLK signal, which may be provided on the connector, may be used to indicate whether data is currently being evaluated. By incrementing the counter each time the FastCLK signal is asserted, the counter will be incremented each evaluation step. Other signals, e.g., provided via other connections than the connectors 24, may be used in the alternative.

The step start signal may be generated, for example, from a DSTEP signal, which is asserted at the beginning of an evaluation cycle. In the alternative, an MSTEP signal provided by the logic emulator via an external cable may be used. The use of an MSTEP signal is described, for example in U.S. S/N 09/523,053 by Cook et al., filed on March 10, 2000, the disclosure of which is incorporated by reference herein. This signal asserts itself at the start of an evaluation cycle and deasserts at the end of an evaluation cycle. In the alternative, virtual logic compiled into a compiler model may be used to manage a dedicated pin that will tell the buffer card to reset its write/read pointers to step 0.

From a review of Fig. 3, several conditions should be appreciated. First, times 7-14 and 20-28 reflect respective pauses during the emulation of clock domains A and B, and illustrate the fact that in the case of a pause in clock domain A, the same data will be read from the XBI pin on successive evaluation cycles in clock domain B (e.g., at times 3-6 and 10-14). Likewise, a pause in clock domain B will result in the data asserted on the XBO pin by clock domain A during one cycle (times 22-28) being discarded before being read.

Moreover, as illustrated at times 29-33, a potential unstable condition may be signaled when it is determined that the transmitter and receiver are attempting to read and write into the buffer at the same time -- resulting in a potentially unstable output. Indication of an unstable condition may be performed, for example, via a separate signal path or another condition on the signal path (e.g., a tri-state condition). Detection and reporting of an unstable condition is useful for debugging purposes; however, correction of the condition is typically not desirable since such a condition could occur in the real world system.

It may therefore be seen that reliable communications may be performed between asynchronous clock domains regardless of the relative clock frequency or alignment of the two clock domains.

Various modifications may be made to the illustrated embodiments without departing from the spirit and scope of the invention. For example, a slower step pulse may be derived off a higher frequency step pulse to meet timing constraints of logic on the asynchronous buffer card. Effectively, data could be transferred every *nth* step pulse. Moreover, other manners of resetting a buffer card's write/read pointers may be used, as may other sources for generating the step pulse and step start signals. Moreover, returning to Fig. 2, it will be appreciated that multiple buffers 50 may be provided on an asynchronous buffer card, e.g,. one for each direction of communication between clock domain A and clock domain B. In the alternative, a unified array may be utilized to support bi-directional communication over a given signal path.

As another alternative, it is not necessary for a signal that is output by one clock domain on a certain evaluation step be read on the corresponding evaluation step for the other clock domain, e.g., through the use of a field programmable interconnect device (FPID) or other device to perform step-address translation at the receiver end. Thus, for example, a signal output by clock domain A during evaluation step 3 for the domain A evaluation cycle could be read by clock domain B during evaluation step 7 for the domain B evaluation cycle. Further, it is not necessary for locations (addresses) in the buffer be directly mapped to corresponding evaluation steps. Thus, for example, the signal output by clock domain A during evaluation step

5 could be stored in the 3rd location in the buffer. Data, as well as a location in the buffer, is associated with an evaluation step when there is a set mapping that exists, despite the fact that the mapping is non-linear or non-monotonic.

In addition, while signals are illustrated as being one-bit binary signals, it will be appreciated that a signal and signal path (and thus a stack and buffer) may have varying binary widths. As another alternative, multiple read ports could be provided with independent control of multiple read pointers, to support multiple receiver clock domains, e.g., as in a broadcasting environment where an asynchronous signal is passed to multiple receivers.

Other modifications will be apparent to one of ordinary skill in the art. Therefore, the invention lies in the claims hereinafter appended.